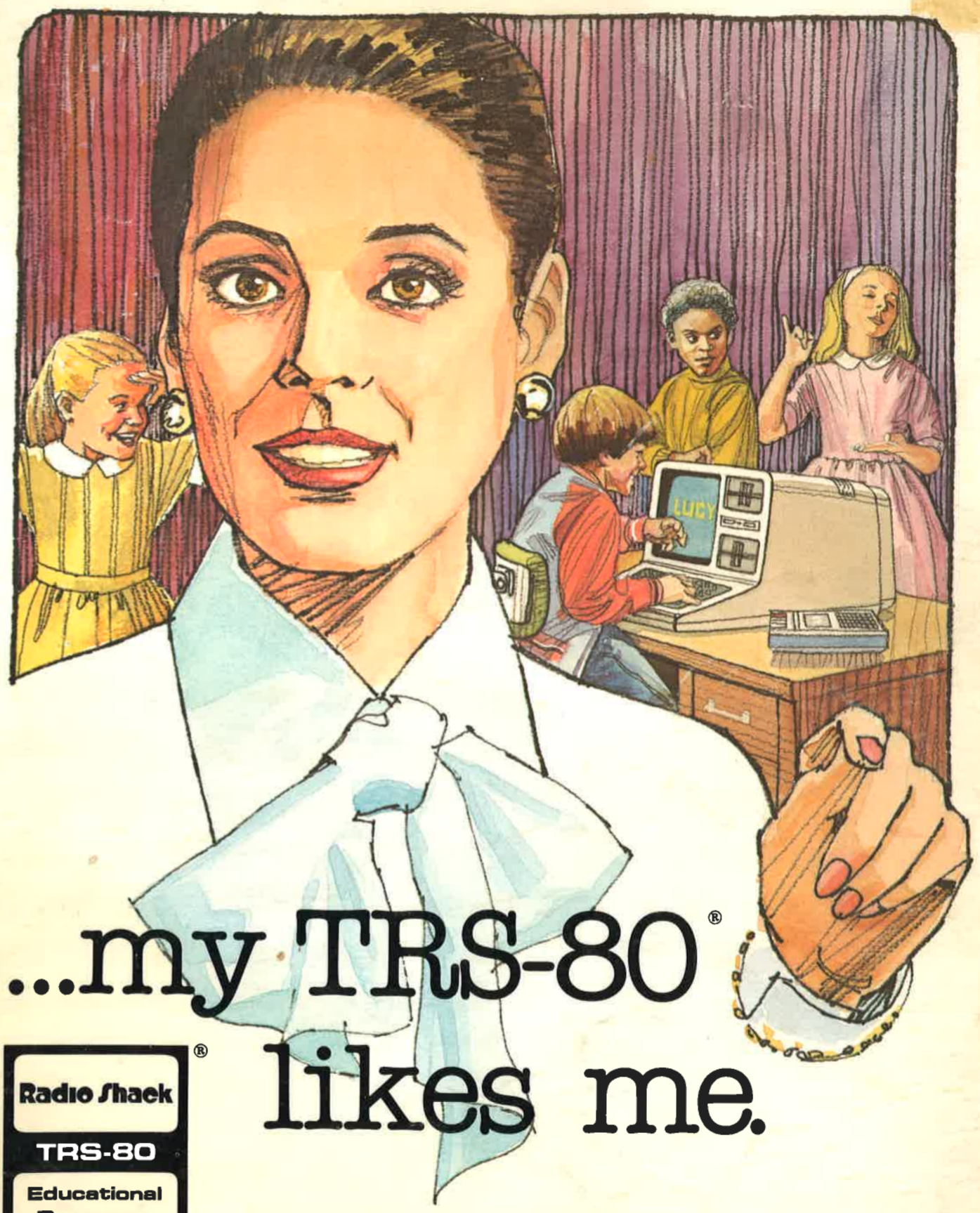


...my TRS-80® likes me.

Radio Shack®

TRS-80

Educational Resource Series

Cat. No. 26-2751

My TRS-80® Likes Me

Bob Albrecht
George Firedrake

Radio Shack®
A DIVISION OF TANDY CORPORATION
FORT WORTH, TEXAS 76102

Illustration on page 18 by Liz Danforth. Reprinted with permission from Flying Buffalo Inc., Scottsdale, Arizona.

Published by Tandy Corporation Under License from Dymax

Contents

A Foreword by

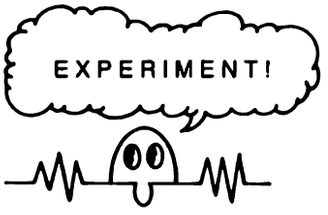
Bob Albrecht

The Radio Shack TRS-80® is a great machine for teaching kids how to use, program and enjoy computers. We've been helping kids learn how to enjoy BASIC for 18 years. For this purpose, the TRS-80 is the best computer we have used, especially if the learners are elementary school children.

IMPORTANT NOTICE! We are not saying that the TRS-80 is the best computer for *all* purposes. We are not saying the TRS-80 is the best overall educational computer. We *are* saying that the TRS-80 is the best computer to use to teach elementary school children how to understand and enjoy BASIC.

So, if you have a TRS-80, grab a bunch of kids and help them learn to understand simple programs in TRS-80 BASIC. Here are ideas on how to go about it, including numerous short programs that many kids have enjoyed.

Think of this booklet as an outline of things to do. *You* explain what is happening, answer questions, invent variations on our ideas and (of course!), use your own ideas. But don't do the typing. Let the kids do the hands-on stuff. Be patient – let them make mistakes, correct their own mistakes and, above all, encourage them to EXPERIMENT!



Oh yes, one more thing. We don't teach people how to program in BASIC. Instead, we help them learn how to read and understand simple BASIC programs. In order to create something in a language, you must first understand the language. That does sound reasonable, doesn't it? Once you understand the language (or some of it), perhaps you can express your own original ideas in the language.

So, help kids (and adults) learn to read and understand BASIC. Presto! Many of them may then teach themselves how to use the language creatively.

magic!

People who don't want to write world shaking original programs might learn something else – computers are sure stupid, and not very mysterious. Or, people might learn to modify someone else's programs to better serve a personal purpose. Or team up with friends who love to write programs in order to create a synergistic program that would not otherwise exist. BASIC, in the universe of home/school/personal computers, is an easy-to-understand starting point for learning about computers.

Begin_

- Show your eager young learners how to hook up the TRS-80 and turn it on.
- Show them how to clear the screen. Start each "episode" with a clear screen.
- Tell them about the prompt (>) and the cursor (–).

When you see this, it is your turn to type.

```
>_
```

- Have someone clear the screen, type her name, then press the ENTER key. The TRS-80 will probably type an error message. Explain that "The TRS-80 doesn't understand you." Never mind – tell them not to worry about occasional misunderstandings; they will soon learn how to make the TRS-80 understand!
- Kids are fascinated by their names. In this booklet, we will do lots of things with names. Start with direct statements such as the following. You will have to remind them several times to press ENTER after typing a statement.

```
PRINT "LUCY"
PRINT "CHARLIE"
PRINT "SNOOPY"
PRINT "LUKE SKYWALKER"
```

And press ENTER

Let every kid do this.

- Now the fun begins. Have each kid enter and run a short three line program that will fill the screen with her name. (Boys are also encouraged to do this.)

```
10 CLS
20 PRINT "LUCY ";
30 GOTO 20
```

Put one or more spaces here

When the above program is RUN, the screen quickly fills with the name 'LUCY.' Then the names seem to move to the right (an optical illusion). Experiment with names and with other characters. Take your time; keep it slow and easy. Let the kids set the pace. Don't try to turn them into computer phreaks. Just let them have fun learning at their own speeds.

- Of course, you will show them how to interrupt a program by pressing the BREAK key. Also show them how to LIST a program and how to change only line 20 to put a different name on the screen.

```
20 PRINT "SNOOPY " ;
```

BEWARE! In Level II BASIC, a syntax error causes the TRS-80 to go into Edit Mode. Press Q to get out– don't get into heavy editing this early in the game.

- Make it easy to change the name. Tell them about "boxes" such as A$, B$, etc., which can hold names or other strings of characters.

```
10 CLS
20 LET A$ = "LUCY "
30 PRINT A$;
40 GOTO 30
```

Try other boxes such as B$, C$, N$ etc. Use single letter variables only; it will save you lots of time. If you are a "computer scientist" or "computer hobbyist" please remember: most of us aren't.

- Still easier. Use the INPUT statement.

```
10 CLS
20 INPUT A$
30 PRINT A$;
40 GOTO 30
```

Let everyone use this program.

- Slow down the action. Use an "empty" FOR-NEXT loop as a time delay as shown in lines 40 and 50 below.

```
10 CLS
20 INPUT A$
30 PRINT A$;

40 FOR Z = 1 TO 100
50 NEXT Z

60 GOTO 30
```

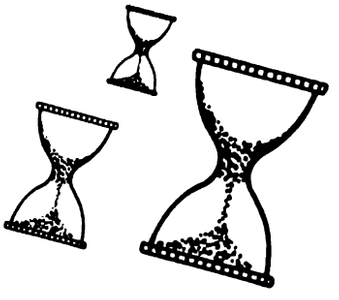
The FOR-NEXT loop (Lines 40 and 50) causes the TRS-80 to count from 1 to 100. Kids are usually surprised at how fast the computer can count to 100. Try other numbers (instead of 100) in the FOR statement.

Slower: Use numbers greater than 100.

Faster: Use numbers less than 100.

- Hmmm ... how about a variable time delay? The following program has one in lines 40, 50 and 60.

```
10 CLS
20 INPUT A$
30 PRINT A$;

40 LET T = 100
50 FOR Z = 1 TO T
60 NEXT Z

70 GOTO 30
```



Change T by replacing line 40 with:

```
40 LET T = 200
```

(Longer delay)

or

```
40 LET T = 50
```

(Shorter delay)

- We can blink a name on and off.

```
10 CLS
20 INPUT A$

30 CLS
40 PRINT A$;
50 LET T = 100
60 FOR Z = 1 TO T
70 NEXT Z

80 CLS
90 LET T = 100
100 FOR Z = 1 TO T
110 NEXT Z

120 GOTO 30
```

Name ON

Name OFF

To slow down the action, make T>100. To speed it up, make T<100.

Slower: `50 LET T = 200`

Faster: `50 LET T = 50`

Also change line 90

- The above program has two identical time delays. Great opportunity to introduce SUBROUTINES!

```
10 CLS
20 INPUT A$

40 CLS
50 PRINT A$;
60 GOSUB 100

70 CLS
80 GOSUB 100

90 GOTO 40

100 LET T = 100
110 FOR Z = 1 TO T
120 NEXT Z
130 RETURN
```

Name ON

Name OFF

Time Delay Subroutine

Take lots of time to explain this one!

- Now might be a good time to show how to put two or more statements on one line.

```
50 FOR Z = 1 TO 100 : NEXT Z
```

Put a colon between statements

```
100 LET T = 100
110 FOR Z = 1 TO T : NEXT Z
120 RETURN
```

- Well, let's not wear out the upper left hand corner of the screen. Tell them about PRINT @ (or PRINT AT in Level I BASIC). We use PRINT @ in line 50 below.

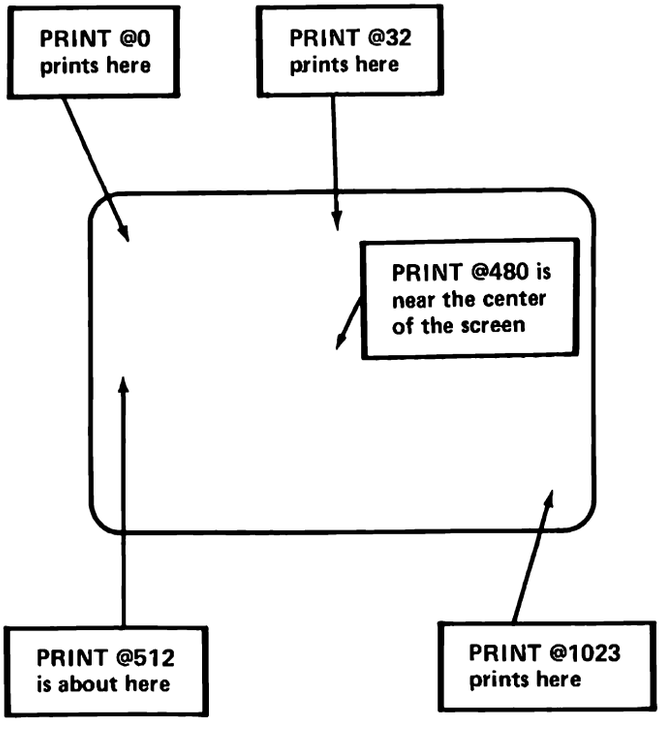
```
10 CLS
20 INPUT A$

40 CLS
50 PRINT @480, A$;
60 GOSUB 100
70 CLS
80 GOSUB 100

90 GOTO 40

100 LET T = 100
110 FOR Z = 1 TO T : NEXT Z
120 RETURN
```

Here is PRINT @

- The above program blinks something near the center of the screen. To blink it somewhere else, try one of these (or invent your own).

```
50 PRINT @0, A$;

50 PRINT @32, A$;

50 PRINT @512, A$;

50 PRINT @1000, A$;
```

PRINT @



Of course, tell them there are 1024 print positions numbered from 0 to 1023. Make a map of the screen showing where the print positions are located. Below is a handy program for putting your name where you want it on the screen. Explain the use of strings within INPUT statements.

```
10 CLS
20 INPUT "WHAT IS YOUR NAME";A$
30 INPUT "WHERE SHALL I BLINK IT";N

40 CLS
50 PRINT @N, A$;
60 GOSUB 100

70 CLS
80 GOSUB 100

90 GOTO 40

100 LET T = 100
110 FOR Z = 1 TO T
120 NEXT Z
130 RETURN
```

Name On at position N

Name OFF (Blank screen)

Time Delay Subroutine

As usual, take your time and make sure everyone understands this program. Remember, they are learning how to read and understand a language. The first step in learning how to create original stuff in a language is to first understand the language.

go slowly...

Our next program causes a name to scamper across the screen from upper left to lower right, then again and again and ... .

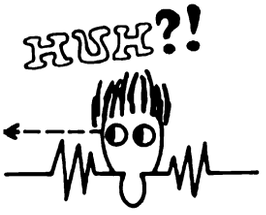
```
10 CLS
20 INPUT "WHAT IS YOUR NAME" ; A$

30 FOR K = 0 TO 1023
40 CLS
50 PRINT @K, A$;
60 NEXT K

70 GOTO 30
```

Too fast? Probably – so, you put in a time delay to slow down the action. We suggest you put it between lines 50 and 60.

Meandering–

So far, we've done "deterministic" things with words on the screen. Now, we will meander – do some random stuff – cause words to wander about the screen, or appear helter-skelter in unpredictable non-patterns.

One of the nicest things about the TRS-80 is the way the RND function works. It gives integer random numbers in a way that is easy to understand and use by people who are not math wizards. Most other BASICs give a random number between 0 and 1. We really can't understand why – most applications (especially games!) require integer random numbers. In most BASICs, much mind boggling math is required to get the desired integer random numbers.



Desired random numbers: 1 or 2
- TRS-80: RND(2)
- Most others: INT(2*RND(1))+1

Desired random numbers: 1, 2 or 3
- TRS-80: RND(3)
- Most others: INT(3*RND(1))+1

Desired random numbers: 1, 2, 3, 4, 5, 6
- TRS-80: RND(6)
- Most others: INT(6*RND(1))+1

Desired random numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- TRS-80: RND(10)–1
- Most others: INT(10*RND(1))

Desired random numbers: 1 to 100
- TRS-80: RND(100)
- Most others: INT(100*RND(1))+1

Got the idea? It goes like this. If n is a positive integer, then RND(n) will give random positive integers in the range, 1 to n. Simple and neat!

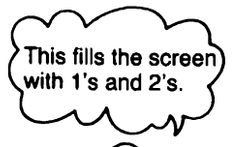
Bonus! RND(0) will give random numbers in the range, 0 to 1, just like all the other computers. The TRS-80 gives you the best of both worlds.

IMPORTANT NOTICE! Remember, this booklet is about teaching BASIC to kids. So, the RND function is one of the most important elements of BASIC. It is one of the things that makes BASIC fun for kids! For ease in teaching, use the integer RND function.

Experiment!

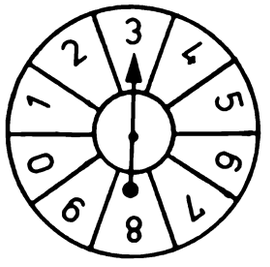
Do some experiments.

```
10 CLS
20 PRINT RND(2);
30 GOTO 20
```

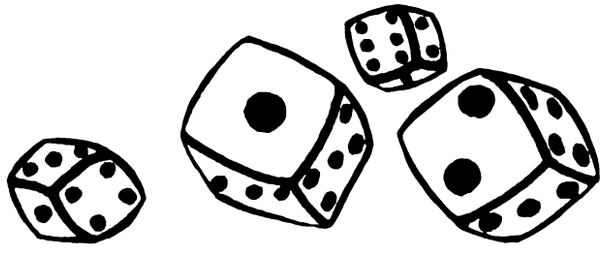



```
10 CLS
20 PRINT RND(3);
30 GOTO 20
```

```
10 CLS
20 PRINT RND (6);
30 GOTO 20
```




```
10 CLS
20 PRINT RND(10)-1;
30 GOTO 20
```



OK, so RND gives random numbers. So what? Well, let's use them to put a name here, there or anywhere on the screen. Here for a moment, there for a moment, somewhere else for a moment.

```
10 CLS
20 X = RND(1023)
30 PRINT @X, "LUCY";
40 T = 100          <-- Time delay
50 FOR Z = 1 TO T : NEXT Z
60 GOTO 10
```

Explain lines 20 and 30 carefully. Line 20 gives a random integer from 1 to 1023, inclusive, and puts this number in box X. Line 30 causes the name "LUCY" to be printed at position X on the screen. Lines 40 and 50 are a time delay. Change Line 40 to decrease or increase the delay.

Faster: Make T<100

Slower: Make T>100

Why let Lucy get all the glory? Let each kid change line 30 to his or her name. To make it easier, try the following program.

```
10 CLS
20 INPUT "WHAT IS YOUR NAME" ; N$
30 CLS
40 X = RND(1023)
50 PRINT @X, N$;
60 T = 100
70 FOR Z = 1 TO T : NEXT Z
80 GOTO 30
```

As usual, slowly and patiently explain what is happening, to your eager young learners. If they say, "What if...?" then you say, "Try it and find out!"

In the spirit of experimentation, try this. Change line 80 to:

```
80 GOTO 40
```

Now RUN the program to find out what happens.

FuturePlay

This material is intended as an outline for parents or teachers on how to help kids learn to use, program and enjoy computers. These ideas are best used when a kid asks, "How does the computer do that?" or "How can I make the computer do what I want it to do?," or "Can the computer tell me (whatever)?" or ... .

So you have a TRS-80 and the kids have played Hurkle, Taipan, Adventure, Taxman, Invasion Force, and countless other games. Now, they want to know more; they want to write game-playing programs; they want to put interesting visual patterns on the screen. THEY WANT TO CONTROL THE COMPUTER.

Why not? They control the future; so, let them control the computer, the tool of the future. Give your kids this tool. Let them shape it in ways unknown to us. Then stand back and enjoy!!

Patterns

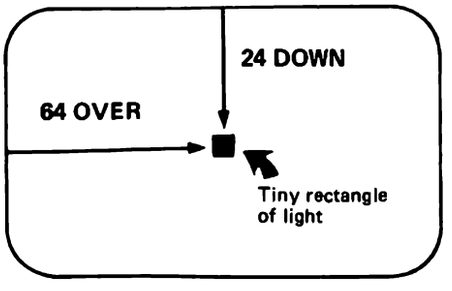
Patterns! ... That's what we will do next. Show you how to put patterns on the screen, using tiny rectangles of light. And, we will do some "computer art," using the familiar RND function and a new BASIC statement called SET.

- SET turns on a tiny rectangle of light somewhere on the screen.

Somewhere on the screen? Well, of course, we must tell the TRS-80 *where* to turn on the tiny rectangle of light.

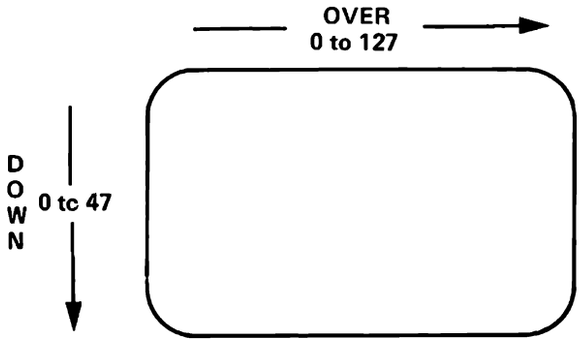
```
SET (64, 24)
```

This far OVER **This far DOWN**



Think of it like this: SET(OVER, DOWN)

- OVER can be a whole number, 0 to 127.
- DOWN can be a whole number, 0 to 47.



Experiment

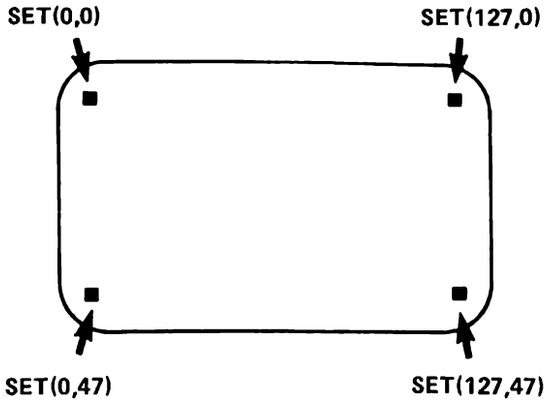
Try these to get the hang of things.

- Press CLEAR, then press ENTER
- Type SET(0,0) and press ENTER
  Where is the tiny rectangle?
- Type SET(127,0) and press ENTER
  Where is the tiny rectangle?
- Type SET(0,47) and press ENTER
  Where is the tiny rectangle?
- Type SET(127,47) and press ENTER
  Where is the tiny rectangle?

Now enter this little program and RUN it.

```
10 CLS

20 SET(0, 0)
30 GOSUB 1000

40 SET(127, 0)
50 GOSUB 1000

60 SET(0, 47)
70 GOSUB 1000

80 SET(127, 47)

90 GOTO 90

1000 T = 500
1010 FOR Z = 1 TO T : NEXT Z
1020 RETURN
```

This is what you will see:



The tiny rectangles at (0,0), (127,0), (0,47) and (127,47) will come on, one at a time, because of the time delay following each SET statement. Change line 1000 to speed up or slow down the action.

For each of the following, first clear the screen. Then type the SET statement. *Before* you press ENTER, guess where the tiny rectangle of light will appear (OK to touch the screen).

Then do it ... press ENTER ... were you close?

CLEAR the screen before you do each of the following.

- Type SET(0, 24), guess, press ENTER
- Type SET(127, 24), guess, press ENTER
- Type SET(64, 0), guess, press ENTER
- Type SET(64,47), guess, press ENTER
- Type SET(64, 24), guess, press ENTER
- Type SET(32, 12), guess, press ENTER

Do you want more practice? If so, try the following program.

```
10 CLS
20 INPUT "HOW FAR OVER"; OVER
30 INPUT "HOW FAR DOWN"; DOWN

40 SET(OVER, DOWN)

50 T = 1000
60 FOR Z = 1 TO T : NEXT Z

70 GOTO 10
```

Use the above program to EXPERIMENT!

TROUBLE! If OVER is less than 0 or more than 127, you will get an error message. If DOWN is less than 0 or more than 47, you will get an error message. It will look like this:

```
?FC ERROR IN 40
```

So, please cooperate. OVER can be any whole number from 0 to 127; DOWN can be any whole number from 0 to 47.

Starfall

Now, pretend that the screen is the sky. Also, pretend that each tiny rectangle of light is a star appearing in the night sky. Make it happen by storing and running this program.

```
100 CLS

110 OVER = RND(127)

120 DOWN = RND(47)

130 SET (OVER, DOWN)

140 GOTO 110
```

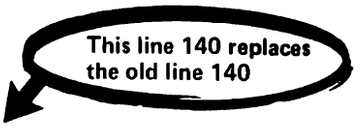
When you RUN this program, the sky will begin filling with stars. Try to find ... constellations, shapes, patterns ... as stars turn on.

Too fast? Add a time delay.

```
133 T = 100

135 FOR Z = 1 TO T

137 NEXT Z
```

Or, here is a trick you can use to stop the action momentarily while you stargaze, then continue letting more stars come out.

Add these lines to the original program.




```
140 IF INKEY$ = "" THEN 110

150 IF INKEY$ = "" THEN 150

160 GOTO 110
```

Oops! INKEY$? Quotation marks with nothing inside? Patience, please. Soon, we will describe INKEY$, a most useful feature of TRS-80 BASIC.

In the meantime, go ahead and use INKEY$. Carefully, type lines 140 and 150 as shown below.

```
140 IF INKEY$ = "" THEN 110
```

No space between quotes.

```
150 IF INKEY$ = "" THEN 150
```

No space between quotes.

Have you added the above lines (140, 150, 160) to the original program? Yes, ... good! Now, RUN the modified program. To stop it, press any key; then, to continue, press any key. Now you can stop time! Or restart it! Happy stargazing.

Walkabout

Suppose you start near the center of the screen (OVER=64, DOWN=24) and wander. Up? Down? Left? Right? Try this program.

```
100 CLS

110 OVER = 64          Start near the center
120 DOWN = 24          of the screen

130 SET (OVER, DOWN)   Turn on a light

140 A = RND(3)-2       A will be –1, 0 or 1
150 B = RND(3)-2       B will be –1, 0 or 1

160 OVER = OVER + A    Well, obviously (?), this
170 DOWN = DOWN + B    will change OVER and
180 GOTO 130           DOWN
```

Dismay not! Try this program. It will start near the center of the screen and ... meander. Eventually, it will (probably) stop with an error message.

```
?FC ERROR IN 130
```

That happens because we tried to wander off the screen. Off the screen? Well, that means: OVER became less than 0 or more than 127 or DOWN became less than 0 or more than 47. More about that in a minute –

In the meantime, try this program.

```
100 CLS
110 X = RND(128)-1      <-- X will mean 'OVER.'
120 Y = RND(48)-1       <-- Y will mean 'DOWN.'
130 SET (X, Y)
140 SET (X, 47-Y)
150 SET (127-X, Y)
160 SET (127-X, 47-Y)
170 GOTO 110
```

Remember: the first number means OVER; the second number means DOWN

Run it for awhile, then press BREAK, RUN it again, etc ...

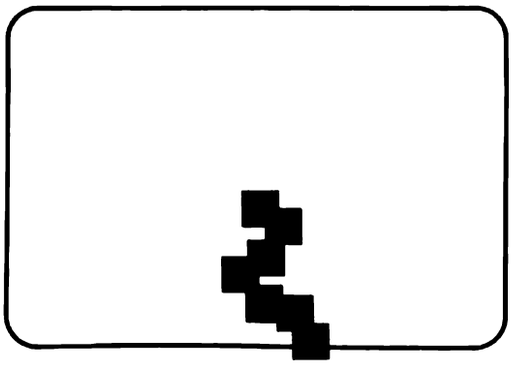
More Walkabout

Here again is our "walkabout" program.

```
100 CLS

110 OVER = 64
120 DOWN = 24

130 SET (OVER, DOWN)

140 A = RND(3)-2
150 B = RND(3)-2

160 OVER = OVER + A
170 DOWN = DOWN + B
180 GOTO 130
```

Remember, it eventually tries to wander off the screen. This happens if OVER<0 or OVER>127 or DOWN<0 or DOWN>47. When this happens, the TRS-80 stops with the following error message:

```
?FC ERROR IN 130
```

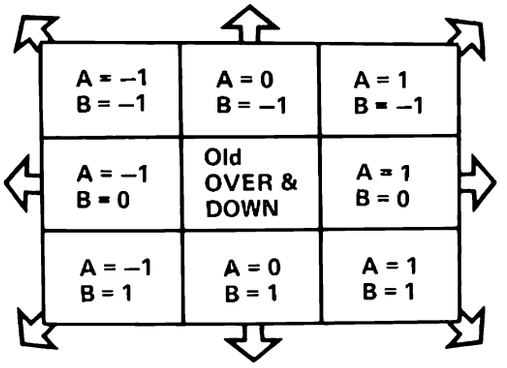


Aha! Let's fix that little bug in our program. Here is our bug tranquilizer. Add it to the program.

```
163 IF OVER<0 THEN OVER=OVER+1
167 IF OVER>127 THEN OVER=OVER-1
173 IF DOWN<0 THEN DOWN=DOWN+1
177 IF DOWN>47 THEN DOWN=DOWN-1
```

These additions to the program will sometimes seem to make the screen image "bounce along" against the edge of the screen. Try it. As the pattern develops, what shapes do you see? A castle? A dragon? A flower? An angular cloud? Buildings on a hillside?

Our tiny rectangle of light can wander in 8 directions.



Wouldn't it be nice to be able to stop the computer and look at the pattern on the screen? Or photograph it? Easy to do. Simply add INKEY$ to our WALKABOUT program.

```
100 CLS

110 OVER = 64
120 DOWN = 24

130 SET(OVER, DOWN)

140 A = RND(3)-2
150 B = RND(3)-2

160 OVER = OVER + A
170 DOWN = DOWN + B

180 IF INKEY$ = "" THEN 130
190 IF INKEY$ = "" THEN 190 ELSE 130
```

Remember: Do NOT type a space between quotation marks.

Now, having added the above lines 180 and 190, RUN the program. To stop it, press any key (except BREAK). Gaze upon the pattern on the screen. Then, press any key (except BREAK) and the pattern will continue growing.

OK, OK. Some of you BASIC users have never seen

IF..THEN...ELSE...

Another plus for TRS-80 BASIC!

```
IF INKEY$ = "" THEN 190 ELSE 130
```

If no one pressed a key, GOTO 190

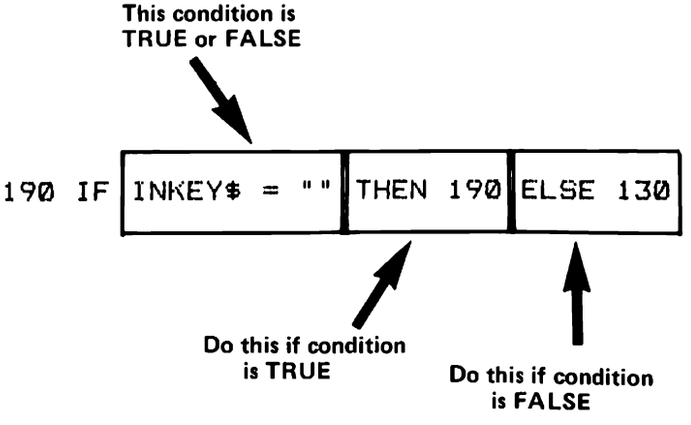
However, if someone did press a key, GOTO 130

The statement:

```
190 IF INKEY$ = "" THEN 190 ELSE 130
```

tells the computer: If INKEY$ is empty (no key has been pressed) then go to line 190. However, if INKEY$ is not empty (a key has been pressed), then (ELSE) go to line 130.

Remember, if no key is pressed, the condition INKEY$="" is *true*. In this case, the TRS-80 obeys the THEN clause and goes to line 190. However, if someone presses a key, the condition INKEY$="" is *false*. The TRS-80 moves on to the ELSE clause and goes to line 130.



Got it? If not, read on — we will go over it again in the next few pages.

The Key to INKEY$

The following program makes random symmetric patterns on the screen.

```
100 REM**MANDALA
110 CLS

200 REM**TURN ON 4 STARS, SYMMETRIC
201 REM**ABOUT CENTER OF SCREEN
210 X = RND(128) - 1
220 Y = RND(48) - 1
230 SET(X, Y)
240 SET(X, 47-Y)
250 SET(127-X, Y)
260 SET(127-X, 47-Y)

300 REM**IF THE 'S' KEY IS NOT
301 REM**PRESSED, GOTO 210
310 IF INKEY$<> "S" THEN 210

400 REM**DO THIS IF THE 'S' KEY
401 REM**WAS PRESSED
410 T = 2000
420 FOR Z = 1 TO T : NEXT Z
430 GOTO 210
```

Enter the program and RUN it. The screen will begin filling with "stars." Press the **S** key. MANDALA pauses for a few seconds, then continues from where it left off, putting more stars on the screen.

Press the **S** key. MANDALA pauses for a few seconds, then continues. Do this several times. When you tire of this, press BREAK to stop the TRS-80.

The INKEY$ function scans the keyboard. If you press a key, the value of INKEY$ will be a one-character string, the key that you pressed.

The statement:

```
310 IF INKEY$<> "S" THEN 210
```

tells the TRS-80

- If the **S** key has NOT been pressed, then go to line 210.
- If the **S** key has been pressed, go to the next line number following line 310.

Following line 310 is a time delay in lines 410 and 420. So, if you press the **S** key, the TRS-80 goes to the time delay. During the time delay, nothing changes on the screen. After the time delay, the TRS-80 moves on to line 430 which sends it back to line 210. More stars appear.

The value of INKEY$ can be one, and only one, character. We could use the Z key to interrupt MANDALA by changing line 310 as follows:

```
310 IF INKEY$<> "Z" THEN 210
```

Or, we could use the space bar:

```
310 IF INKEY$<> " " THEN 210
```

Put a space here

Or, by using the following sneaky trick, we can press almost *any* key to interrupt MANDALA.

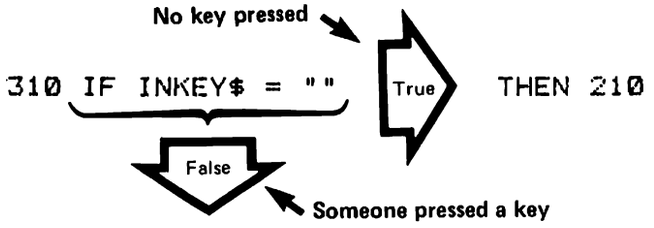
```
310 IF INKEY$ = "" THEN 210
```

Huh? There is *nothing* between quotation marks!

This says: If the value of INKEY$ is "nothing" then go to line 210. This will happen if *no* key has been pressed.

If someone presses a key, the value of INKEY$ will be "something." In this case, the condition INKEY$="" will be *false* and the TRS-80 will go to the line following line 310.

FOLLOW THE ARROWS



Remember, in line 310 there is *nothing* between quotation marks. Since we use quotation marks to enclose strings, you can think of "" as the *empty string.*

MATH TEACHERS: Compare the *empty string* with the *empty set.*

With the above line 310, almost any key will interrupt the action on the screen. Exceptions are the BREAK key, which stops the program, and the SHIFT key. By itself, the SHIFT key does nothing. However, if you hold the SHIFT key down and press most any other key, you will get the interrupt and time delay.

Our time delay is a few seconds. If you wish to gaze at a pattern for a longer time, increase the value of T in line 410. Or, rewrite everything from line 300 on, as follows:

```
300 REM**IF NO KEY PRESSED, GOTO 210
310 IF INKEY$ = "" THEN 210

400 REM**WAIT FOR KEY, THEN GOTO 210
410 IF INKEY$ = "" THEN 410
420 GOTO 210
```

Line 410 tells the TRS-80 to keep scanning the keyboard until someone presses a key. If someone presses a key, the TRS-80 drops down to line 420, then goes back to line 210.

Try it. Enter the changes, type RUN and watch MANDALA. If you see a pattern that you like, press most any key and gaze for as long as you wish. Then, press most any key — MANDALA will continue.

Before we leave MANDALA, let's try one more thing. We can replace lines 410 and 420 with a single line.

Instead of:

```
410 IF INKEY$ = "" THEN 410
420 GOTO 210
```

Do this:

```
410 IF INKEY$ = "" THEN 410 ELSE 210
```

Our new line 410 tells the TRS-80:

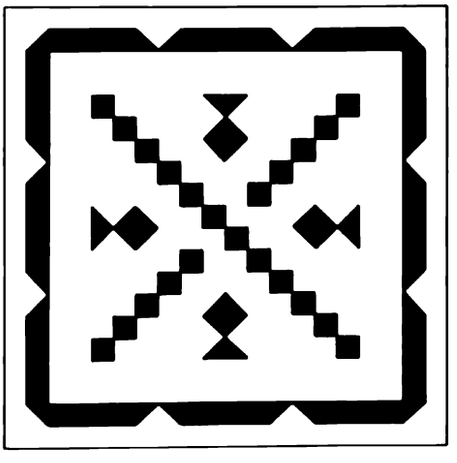
- If no key has been pressed, then go to line 410.
- Else (a key has been pressed), go to line 210.

In general, the IF-THEN-ELSE statement looks like this.

IF condition THEN statement ELSE statement

Do this statement if condition TRUE.

Do this statement if condition FALSE.



Interlude–A Game

Once upon a time, computers were veerrryyyy expensive. Many of them "talked" to people by means of a rather large, and definitely noisy, device called a ... (suspense) ... Teletype®!

The Teletype (or TTY, as it became affectionately known) typed information on paper. From these almost prehistoric times, we bring you a simple computer game enjoyed by multitudes of children – a game called NUMBER or, if you prefer, GUESS MY NUMBER. Our version of NUMBER, shown below, is modified slightly (but only slightly) for TRS-80.

```
100 REM**NUMBER, A NUMBER GUESSING GAME
110 RANDOM

200 REM**TELL PLAYER ABOUT THE GAME
210 CLS
220 PRINT "I AM THINKING OF A NUMBER
    FROM 1 TO 100."
230 PRINT "GUESS MY NUMBER!!!"

300 REM**COMPUTER 'THINKS' OF A NUMBER, X
310 X = RND(100)

400 REM**GET GUESS, G, COMPARE WITH X
410 INPUT "YOUR GUESS"; G
420 IF G=X THEN 610
430 IF G<X THEN PRINT "TRY BIGGER"
440 IF G>X THEN PRINT "TRY SMALLER"
450 GOTO 410

600 REM**WINNER!
610 PRINT "THAT'S IT!!! MY NUMBER IS" X
620 FOR T = 1 TO 200 : NEXT T
630 GOTO 210
```

Well, try the above program. It uses the screen abysmally, doesn't it? This program is fine for printed output, but terrible for TV. The printer is a one-dimensional medium; the TV screen is a two-dimensional medium.

IMPORTANT NOTICE: When you enter the above program, type line 220 on a single line. We broke it into two lines to squeeze it into the available space.

```
220 PRINT "I'M THINKING OF A NUMBER
    FROM 1 TO 100."
```

We prefer the following version of NUMBER.

```
100 REM**GUESS MY NUMBER
110 RANDOM

200 REM**TELL ABOUT THE GAME
210 CLS
220 PRINT@333, "I'M THINKING OF A NUMBER
    FROM 1 TO 100"
230 PRINT@600, "GUESS MY NUMBER"
240 T = 2000
250 FOR Z = 1 TO T : NEXT Z

300 REM**X IS SECRET NUMBER, C IS
    GUESS COUNTER
310 X = RND(100)
320 C = 0
330 CLS

400 REM**GET GUESS (G) ADD 1 TO COUNTER
410 PRINT@960, "YOUR GUESS"; : INPUT G
420 C = C + 1

500 REM**COMPARE GUESS AND SECRET NUMBER
510 IF G=X THEN 610
520 IF G<X PRINT@896+32, "TRY BIGGER"
530 IF G>X PRINT@896+32, "TRY SMALLER"
540 GOTO 410

600 REM**WINNER! WATCH THE STARS COME OUT
610 CLS
620 FOR STAR = 1 TO 100
630   PRINT@ RND(1022), "*";
640 NEXT STAR

700 REM**TELL PLAYER SHE WON
710 PRINT@406, "MY NUMBER IS" X
720 PRINT@529, "YOU GOT IT IN" C "GUESSES!!!"
730 PRINT@656, "TO PLAY AGAIN, PRESS ANY KEY"
740 IF INKEY$="" THEN 740 ELSE 210
```

Our second NUMBER program puts the hint, TRY BIGGER or TRY SMALLER, on the same line as the player's guess. Try the program, then study lines 410, 520 and 530 to see how it works. Here are some hints.

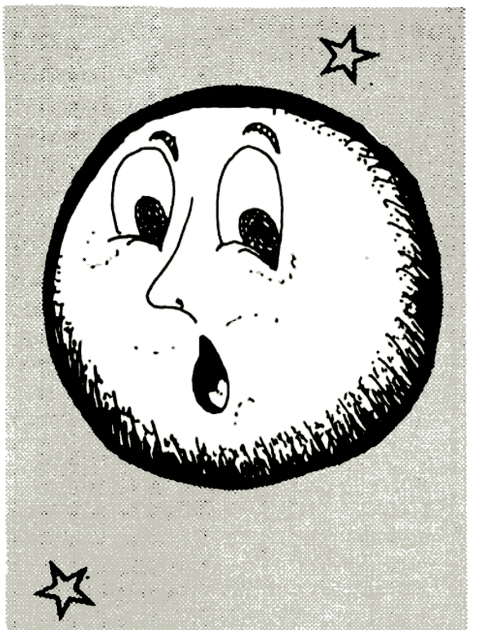
- In line 410, print position 960 is the left edge of the bottom line on the screen.
- If we type something on the bottom line of the screen and then press the ENTER key, everything on the screen will move (scroll) up one place.
- In lines 520 and 530, print position 896 is the left edge of the line just above the bottom line on the screen. Print position 896 + 32 is about half way across that line.

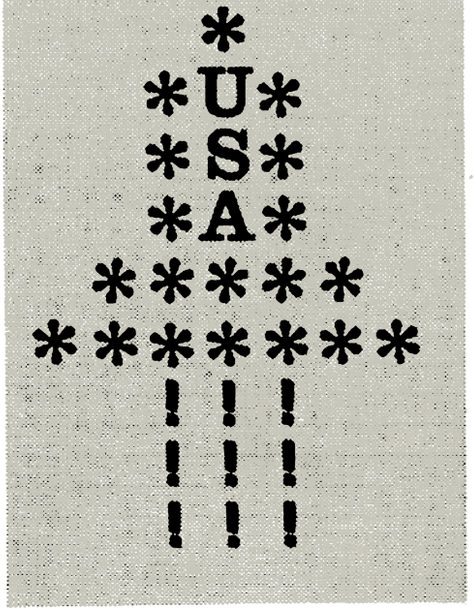
Countdown-Blastoff!

We call the following program COUNTDOWN-BLASTOFF! Enter it in your TRS-80 and RUN it. Note the use of the variable time delay subroutine to show the spaceship "accelerating" off the launch pad (lines 410, 420 and 430).

```
100 REM**COUNTDOWN-BLASTOFF!
110 CLS

200 REM**COUNTDOWN FROM 10 TO 0
210 FOR C = 10 TO 0 STEP -1
220 PRINT C
230 T = 300 : GOSUB 910
240 NEXT C
250 PRINT "BLASTOFF!!!"
260 T = 400 : GOSUB 910

300 REM**SHOW SPACESHIP ON LAUNCH PAD
310 CLS
320 PRINT @512,"    *    "
330 PRINT       "   *U*   "
340 PRINT       "   *S*   "
350 PRINT       "   *A*   "
360 PRINT       "  *****  "
370 PRINT       "*********"
380 T = 400 : GOSUB 910

400 REM**LAUNCH THE SPACESHIP
410 PRINT       "   !!!   " : T=300 : GOSUB 910
420 PRINT       "   !!!   " : T=200 : GOSUB 910
430 PRINT       "   !!!   " : T=100 : GOSUB 910
440 FOR K = 1 TO 16
450 PRINT : T = 100 : GOSUB 910
460 NEXT K

500 REM**ANNOUNCE A SUCCCESSFUL LAUNCH AND STOP
510 CLS
520 PRINT "ALL SYSTEMS ARE GO. EVERYTHING IS AOK!"
530 END

900 REM**TIME DELAY SUBROUTINE
910 FOR Z = 1 TO T : NEXT Z
920 RETURN
```

Line 320 starts the spaceship about halfway down the screen. Type lines 320 through 370 very carefully.





Now that you are launched into space, play our simple reaction time game to pass time until you reach your destination.

```
100 REM**REACTION TIME PROGRAM
110 CLS
120 PRINT "HOW FAST ARE YOU?  I WILL COUNT"
130 PRINT "NEAR THE MIDDLE OF THE SCREEN."
140 PRINT "PRESS THE SPACE BAR TO STOP ME."
150 PRINT
160 PRINT "PRESS ANY KEY AND I'LL BEGIN."
170 IF INKEY$ = "" THEN 170

200 REM**CLEAR SCREEN FOR A RANDOM TIME
210 CLS
220 T = RND(2000)
230 FOR Z = 1 TO T : NEXT Z

300 REM**START COUNTING. SPACE STOPS IT.
310 X = 1
320 PRINT @ 480, X
330 IF INKEY$ <> " " THEN X = X + 1 : GOTO 320

400 REM**PLAYER PRESSED SPACE. PAUSE, REPLAY.
410 T = 2000
420 FOR Z = 1 TO T : NEXT Z
430 GOTO 110
```

Play several times. An average of 10 is fast. Congratulations! If your average is more than 20, well ... maybe you are thinking about something else.

We played the game several times and discovered a way to cheat. We can stop the computer with a count of 1 every time! We can do this, *not* because we are that fast, but because there is a flaw in the program.

Beat the computer! Figure out how to stop the computer at 1 every time, just by pressing the space bar.

IMPORTANT NOTICE! This computer error is not the fault of the computer. Rather, as are almost all computer errors, it is the fault of the programmer! Can you fix the error?

Have you figured out how to beat the REACTION TIME program? Here's how. When the TRS-80 prints PRESS ANY KEY AND I'LL BEGIN, you press the space bar *twice*. Or, press any key, then press the space bar immediately, *before* the computer starts counting near the middle of the screen.

Fix it like this – add the following line to the program:

```
240 X$ = INKEY$
```

This will clear out the value of INKEY$ just before the computer begins counting in lines 310 through 330. Line 240 will be done so fast that it is unlikely that you can press the space bar while the TRS-80 is going from line 240 to line 310.

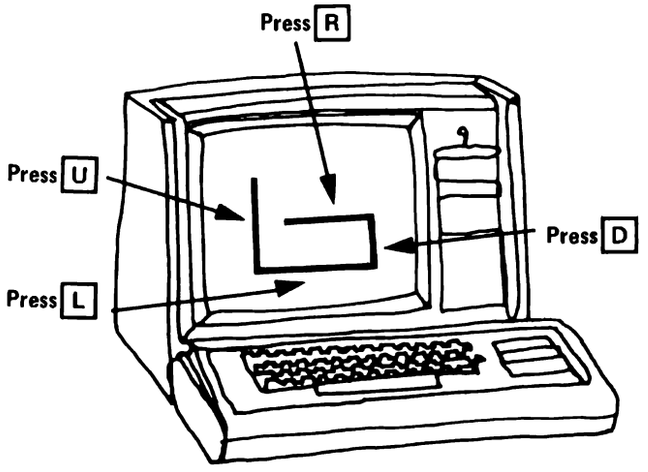
Simple Scribble

The following program will let you scribble on the screen.

```
100 REM**SCRIBBLE ON SCREEN
110 CLS
120 OVER = 64 : DOWN = 24

200 REM**TURN ON A TINY LIGHT
210 SET(OVER, DOWN)

300 REM**WAIT FOR SOMEONE TO PRESS A KEY
310 K$=INKEY$ : IF K$="" THEN 310

400 REM**IF KEY WAS R OR L, CHANGE OVER
410 IF K$="R" THEN OVER = OVER + 1
420 IF K$="L" THEN OVER = OVER - 1

500 REM**IF KEY WAS D OR U, CHANGE DOWN
510 IF K$="D" THEN DOWN = DOWN + 1
520 IF K$="U" THEN DOWN = DOWN - 1

600 GOTO 210
```

When you RUN this program, a single tiny light will come on near the center of the screen. Press the 'R' key a few times. The TRS-80 moves the light to the right, and paints a line along the way.

Then press 'D' a few times. The line prints downward. OK, try 'L' for left and 'U' for up. Got it? Good ... scribble away.



Beware! If you go too far right or too far left, or too far up or too far down, you will get an ? FC ERROR IN LINE 210. We will fix that someday, sometime. Or, better yet, *you* can fix it!

Happy Birthday

Here is a little program which will give you more practice in using SET and PRINT @. We call it a HAPPY BIRTHDAY CARD FOR MOTHER. We began this one by sketching the desired birthday card on a *TRS-80 Video Worksheet*. These worksheets are available from your friendly Radio Shack store. We recommend them – they save *lots of time* in planning how things will look on the screen.

We want our HAPPY BIRTHDAY CARD FOR MOTHER to look like our sketch below.

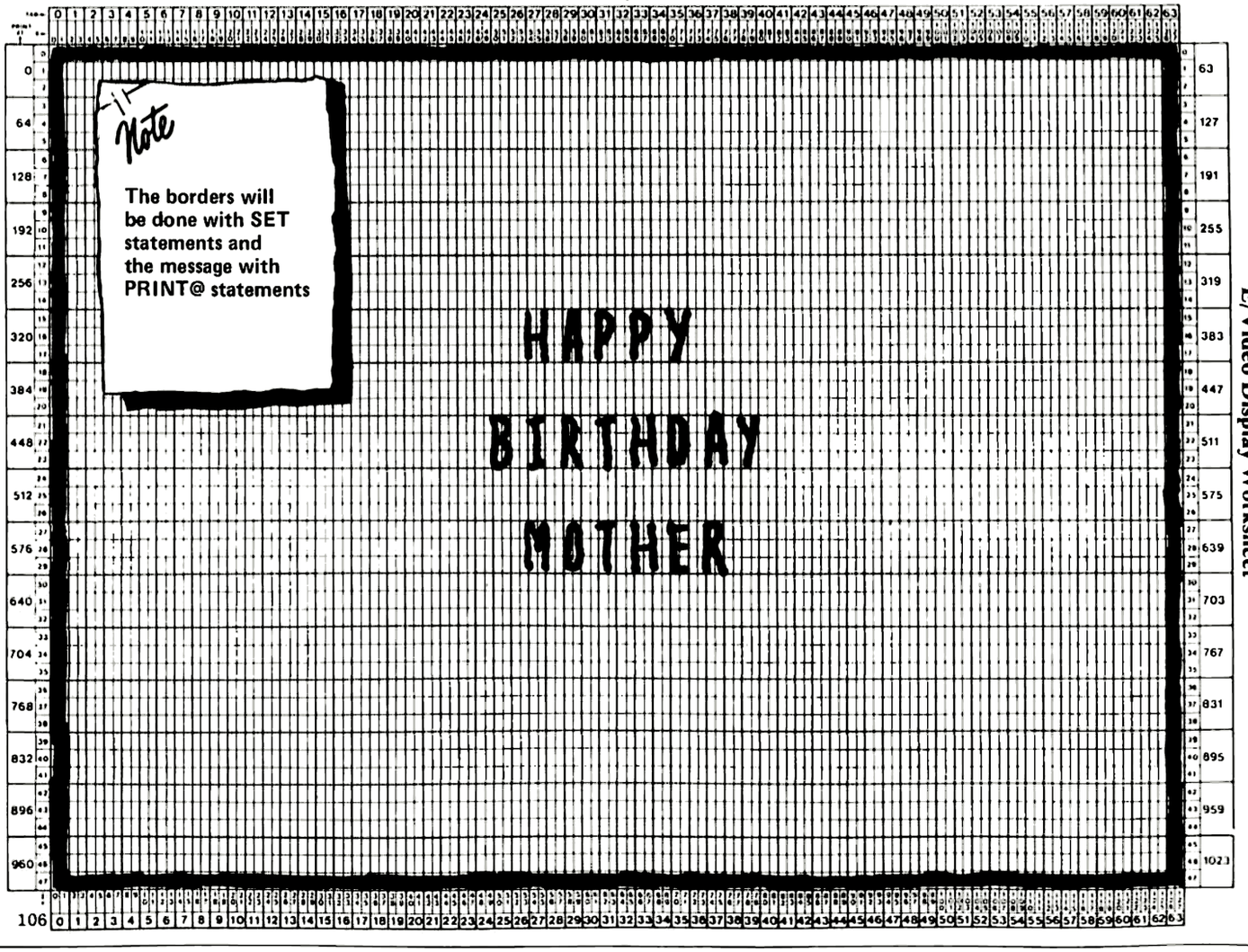
You, of course, can change the message to MERRY CHRISTMAS, SANTA or HAPPY UNBIRTHDAY, GANDALF or ???

Or change lines 210 and 310, as follows.

```
210 FOR OVER = 0 TO 127 STEP 2

310 FOR DOWN = 0 TO 47 STEP 2
```

Your ability to change this program is, of course, your hallmark of excellence (oops! sorry).

```
100 REM**A BIRTHDAY CARD FOR MOTHER
110 CLS

200 REM**DRAW LINES, TOP & BOTTOM
210 FOR OVER = 0 TO 127
220   SET(OVER, 0)
230   SET(OVER, 47)
240 NEXT OVER

300 REM**DRAW LINES, LEFT & RIGHT
310 FOR DOWN = 0 TO 47
320   SET(0, DOWN)
330   SET(127, DOWN)
340 NEXT DOWN

400 REM**THE HAPPY BIRTHDAY MESSAGE
410 PRINT @347, "H A P P Y";
420 PRINT @473, "B I R T H D A Y";
430 PRINT @603, "M O T H E R";

500 REM**WAIT A BIT, THEN REPEAT
510 T = 2000
520 FOR Z = 1 TO T : NEXT Z
530 GOTO 110
```



E/Video Display Worksheet

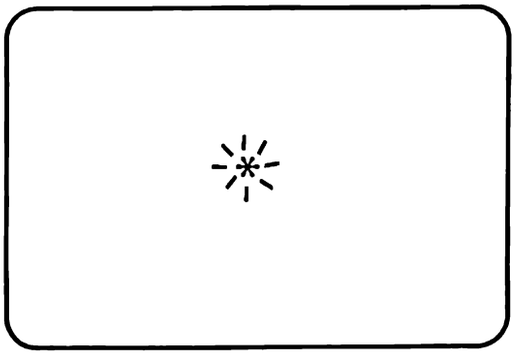
Twinkling Stars

Try these little programs with kids. If someone asks, explain how and why the programs work.

```
100 REM**ONE STAR TWINKLING
110 CLS

200 REM**TWINKLE ON
210 PRINT @480, "*";
220 GOSUB 510

300 REM**TWINKLE OFF
310 PRINT @480, " ";
320 GOSUB 510

400 REM**GOTO 'TWINKLE ON'
410 GOTO 210

500 REM**RANDOM TWINKLE TIME
510 T = RND(1000)
520 FOR Z = 1 TO T : NEXT Z
530 RETURN
```

The twinkle time is random (line 510) and can vary from almost nothing to a couple of seconds. If you want a constant twinkle time, set T to a fixed number in line 510.

The star will twinkle near the center of the screen.

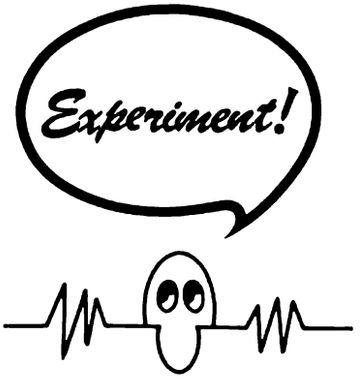


To make it twinkle elsewhere, change lines 210 and 310.

Next, we put 100 stars on the screen, then twinkle them.

```
100 REM**TWINKLE, TWINKLE LOTS
    OF STARS
110 DIM SP(100)

200 REM**PUT STAR POSITIONS IN
    ARRAY SP
210 FOR K = 1 TO 100
220   SP(K) = RND(1022)
230 NEXT K

300 REM**TURN ON STARS
310 CLS
320 FOR K = 1 TO 100
330   PRINT @SP(K), "*";
340 NEXT K

400 REM**PICK A STAR AT RANDOM
410 R = RND(100)

500 REM**TWINKLE OFF
510 PRINT @SP(R), " ";
520 GOSUB 810

600 REM**TWINKLE ON
610 PRINT @SP(R), "*";
620 GOSUB 810

700 REM**GOTO 'PICK A STAR'
710 GOTO 410

800 REM**TWINKLE TIME SUBROUTINE
810 T = 50
820 FOR Z = 1 TO T : NEXT Z
830 RETURN
```

Why do we twinkle a star OFF (line 510) before we twinkle it ON (line 610)? Try it the other way around and see what happens.



Constellations

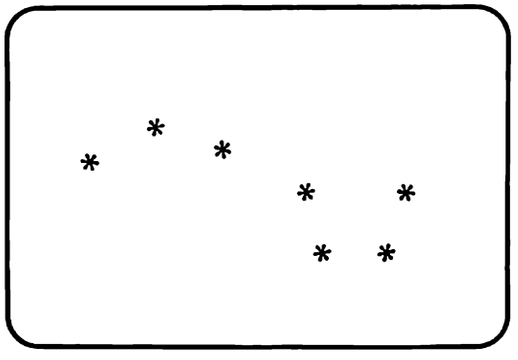
Look at the night sky. Are the stars sprinkled randomly? Perhaps, but from ancient times people saw, and named, patterns called *constellations.*

So, let's put a constellation on the screen. And, since constellations are made of stars, and stars twinkle, let's twinkle our constellation's stars.

Well, we don't know what constellation you would like to see, so we will let *you* enter it. Here is what happened when we ran the program.

```
HOW MANY STARS? 7
?392
?337
?411
?547
?559
?748
?740
```

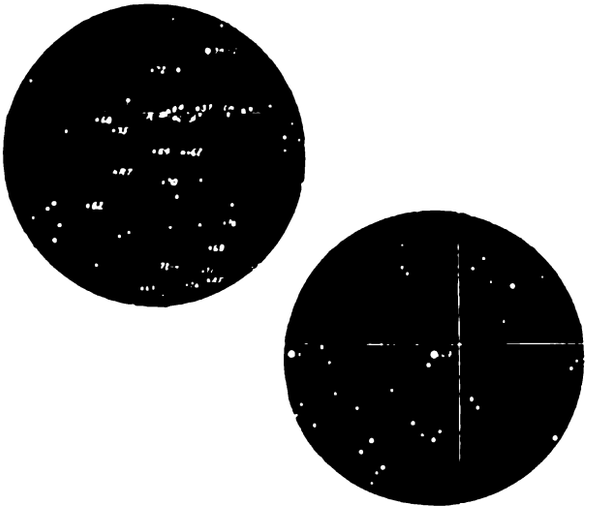
First, we told the TRS-80 that our constellation had seven (7) stars. It then typed seven question marks. After each question mark, we entered the screen position of a star, then pressed ENTER. After entering the 7th star, this is what we saw.



And, as we watched, the stars twinkled.

Here is the program.

```
100 REM**TWINKLING CONSTELLATIONS
110 DIM SP(100)

200 REM**GET STAR PLACES
210 CLS
220 INPUT "HOW MANY STARS"; N
230 FOR K = 1 TO N
240   INPUT SP(K)
250 NEXT K

300 REM**TURN ON STARS
310 CLS
320 FOR K = 1 TO N
330   PRINT @SP(K), "*";
340 NEXT K

400 REM**PICK A STAR AT RANDOM
410 R = RND(N)

500 REM**TWINKLE OFF
510 PRINT @SP(R), " ";
520 GOSUB 810

600 REM**TWINKLE ON
610 PRINT @SP(R), "*";
620 GOSUB 810

700 REM**GOTO 'PICK A STAR'
710 GOTO 410

800 REM**TWINKLE TIME SUBROUTINE
810 T = 50
820 FOR Z = 1 TO T : NEXT Z
830 RETURN
```



Gamemaster's Dice

Have you heard of *Dungeons and Dragons, Runequest* or *Tunnels and Trolls*? These are fantasy role-playing games. If you are a teacher and haven't heard about these games, ask your students about them. In the unlikely event that they can't fill you in, write to the following companies for information.

Dungeons and Dragons (D&D) from TSR Hobbies
P.O. Box 756, Lake Geneva, WI 53147

Runequest (RQ) from The Chaosium
P.O. Box 6302, Albany, CA 94706

Tunnels and Trolls (T&T)from Flying Buffalo, Inc.
P.O. Box 1467, Scottsdale, AZ 85252

To play any of these games, you must create one or more characters, then guide your character(s) through adventures in a universe created by a gamemaster. To create a character, you will roll three six-sided dice several times. We will use Runequest as an example.

A Runequest character has seven characteristics: strength (STR), intelligence (INT), power (POW), constitution (CON), dexterity (DEX), charisma (CHA) and size (SIZ).

These characteristics determine a character's ability to use weapons, fight, learn and use magic, sustain damage, lead others, and so on.

Each characteristic is determined by rolling three six-sided dice. So a characteristic can range from a low of 3 to a high of 18.

Here is a program to create a Runequest character.

```
100 REM**CREATE A RUNEQUEST CHARACTER
110 CLS

200 REM**ROLL  3 DICE 7 TIMES
210 GOSUB 310 : PRINT "STR",DICE
220 GOSUB 310 : PRINT "INT",DICE
230 GOSUB 310 : PRINT "POW",DICE
240 GOSUB 310 : PRINT "CON",DICE
250 GOSUB 310 : PRINT "DEX",DICE
260 GOSUB 310 : PRINT "CHA",DICE
270 GOSUB 310 : PRINT "SIZ",DICE
280 PRINT
290 STOP

300 REM**SUBROUTINE TO ROLL 3 DICE
310 D1 = RND(6)
320 D2 = RND(6)
330 D3 = RND(6)
340 DICE = D1 + D2 + D3
350 RETURN

999 END
```

Let's roll a character. We type RUN, press ENTER and ...

```
STR              15
INT              7
POW              9
CON              12
DEX              10
CHA              11
SIZ              13
BREAK IN 290
READY
>-
```

We have our first RQ character. He is big (SIZ = 13), strong (STR = 15), above average in soaking up damage (CON = 12) and has average dexterity (DEX = 10). He is also not too bright (INT = 7) and has below average power (POW = 9). He will *not* become a magic user. Looks like our character would be a good fighter.

Let's roll another character. Again, we type RUN and press ENTER.

```
STR              11
INT              16
POW              17
CON              10
DEX              11
CHA              17
SIZ              11
BREAK IN 290
READY
>-
```

A most unlikely person! About average in strength, constitution, dexterity and size. But, look at intelligence (16), power (17) and charisma (17)! This character must be a wizard destined to lead.

The Tunnels and Trolls character has six characteristics: strength (STR), intelligence (IQ), luck (LK), constitution (CON), dexterity (DEX) and charisma (CHR). We could modify our Runequest program to get a program to create a T&T character. Instead, we will show you a completely different way to do this in the program below.

The six T&T characteristics are in the DATA statement (line 910) followed by an end of data flag, ZZZ. Characteristics are read, one at a time, by the READ statement in line 220. For each characteristic the TRS-80 "rolls" three dice, then prints the characteristic and its numerical value.

Eventually, the TRS-80 reads the end of data flag (ZZZ) and, thanks to line 230, goes to line 310. The rest is up to you. If you press the space bar, the computer will start over from line 110.

Hmmm... how can you change the program to get a Runequest character?

In Dungeons and Dragons, the characteristics are strength (STR), intelligence (INT), wisdom (WIS), constitution (CON), dexterity (DEX) and charisma (CHA). How can you change the program to get a D&D character?


```
100 REM**CREATE A T&T CHARACTER
110 CLS

200 REM**ROLL THE CHARACTERISTICS
210 RESTORE
220 READ CH$
230 IF CH$ = "ZZZ" THEN 310
240 D1 = RND(6)
250 D2 = RND(6)
260 D3 = RND(6)
270 DICE = D1 + D2 + D3
280 PRINT CH$, DICE
290 GOTO 220

300 REM**PRESS 'SPACE' TO DO AGAIN
310 PRINT
320 PRINT "PRESS THE SPACE BAR TO GET
    ANOTHER CHARACTER"
330 K$ = INKEY$ : IF K$ = "" THEN 330
340 IF K$ = " " THEN 110 ELSE 330

900 REM**CHARACTERISTICS
910 DATA STR,IQ,LK,CON,DEX,CHR,ZZZ
```

This statement checks for the "end of data" flag.

ZZZ is the "end of data" flag. See line 230.

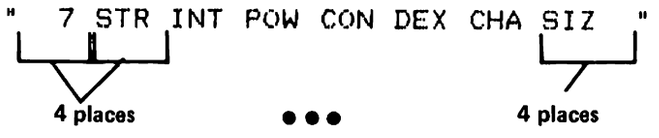
Instead of a separate program for each game, let's write one program to create a character in any of the three game systems. The following table shows the names of characteristics in the three systems. We also show the abbreviations for each characteristic in parentheses.

D&D	RQ	T&T
Strength (STR)	Strength (STR)	Strength (STR)
Intelligence (INT)	Intelligence (INT)	Intelligence (IQ)
Wisdom (WIS)	Power (POW)	Luck (LK)
Constitution (CON)	Constitution (CON)	Constitution(CON)
Dexterity (DEX)	Dexterity (DEX)	Dexterity (DEX)
Charisma (CHA)	Charisma (CHA)	Charisma (CHR)
	Size (SIZ)	

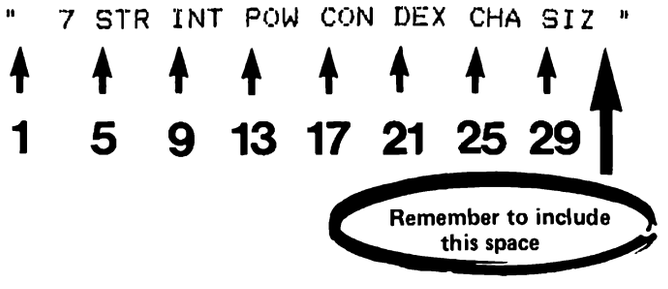
OK, let's do this program, piece by piece. We begin like this.

```
100 REM**CREATE A FANTASY CHARACTER FOR
110 REM**D&D, RUNEQUEST, OR T&T
120 CLEAR 200
130 REM**SET UP CHARACTER STRINGS
140 DD$ = "  6 STR INT WIS CON DEX CHA "
150 RQ$ = "  7 STR INT POW CON DEX CHA SIZ "
160 TT$ = "  6 STR IQ  LK  CON DEX CHR "
```

In lines 140 through 160, each string contains the number of characteristics and the abbreviations for the characteristics for one type of game. Within each string, we use exactly four positions to hold each individual substring. For example,



In line 150, the number 7 is in character positions 1 through 4 (space, space, 7, space). STR is in positions 5 through 8 (S, T, R, space), INT is in positions 9 through 12, and so on.



The number 7, in positions 1 through 4, is the number of characteristics in the string. Note the 6 in strings DD$ and TT$.

Let's move on.

```
200 REM**TELL ABOUT THE PROGRAM
210 CLS
220 PRINT "I CAN CREATE A CHARACTER FOR"
230 PRINT
240 PRINT "  DUNGEONS AND DRAGONS (DD)"
250 PRINT "  RUNEQUEST           (RQ)"
260 PRINT "  TUNNELS AND TROLLS  (TT)"
270 PRINT
280 INPUT "WHICH DO YOU WANT (DD,RQ, OR TT)";
    GAME$
```

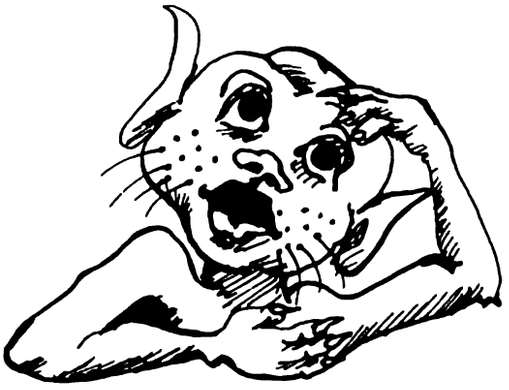
We now expect someone to enter DD, RQ or TT. If anything else is entered, we will start over at line 270.

```
300 REM**CHECK OUT THE VALUE OF GAME$
310 IF GAME$ = "DD" THEN CH$ = DD$ : GOTO 410
320 IF GAME$ = "RQ" THEN CH$ = RQ$ : GOTO 410
330 IF GAME$ = "TT" THEN CH$ = TT$ : GOTO 410
340 PRINT "I DON'T UNDERSTAND" GAME$ : GOTO 270
```

If someone enters DD, RQ or TT, the TRS-80 moves on to line 410 after first setting CH$ to the appropriate characteristic string. If something other than DD, RQ or TT is entered, the TRS-80 says it doesn't understand and repeats the question. So let's see what happens next.

Beginning at line 400, things get a little stickier.

```
400 REM**ROLL THE CHARACTER
410 NC$ = LEFT$(CH$, 4)
420 NC = VAL(NC$)
430 FOR K = 1 TO NC
440    CH = 4*K + 1
450    DICE = RND(6) + RND(6) + RND(6)
460    PRINT MID$(CH$, CH, 4), DICE
470 NEXT K
```



Remember, CH$ is set to the appropriate characteristic string in lines 310 through 330. For example, suppose we enter "RQ" in response to line 280. CH$ will be set equal to RQ$ in line 320.

CH$ = " 7 STR INT POW CON DEX CHA SIZ "

In this case NC$ will become " 7 " in line 410. Thus, NC becomes 7 in line 420. The FOR-NEXT loop in lines 430 through 470 will be done for K = 1 to 7.

K=1

CH = 4*K + 1 = 4*1 + 1 = 5

DICE will be a random number from 3 to 18

MID$(CH$, CH, 4) = MID$(CH$, 5, 4)

= "STR "

So, the computer will print the string "STR " followed by the value of DICE.

K=2

CH = 4*K + 1 = 4*2 + 1 = 9

DICE will be a random number from 3 to 18

MID$(CH$, CH, 4) = MID$(CH$, 9, 4)

= "INT "

So, the computer will print the string "INT " followed by the value of DICE. And so on up to K = 7.

We assume that you understand the rest of the program shown below. True?

```
500 REM**ASK IF SOMEONE WANTS ANOTHER CHARACTER
510 PRINT
520 PRINT "FOR ANOTHER CHARACTER, PRESS THE
    SPACE BAR";
530 K$ = INKEY$ : IF K$ = "" THEN 530
540 IF K$ = " " THEN 210 ELSE 530

999 END
```

Well, there is always another way to do it. Can you complete this program to create a character for DD, RQ or TT?

```
100 REM**CREATE A FANTASY CHARACTER
110 REM**FOR D&D, RUNEQUEST OR T&T
120 CLEAR 200
130 REM**SET UP CHARACTER STRINGS
140 DD$ = "STRINTWISCONDEXCHA"
150 RQ$ = "STRINTPOWCONDEXCHASIZ"
160 TT$ = "STRIQ LK CONDEXCHR"

200 REM**TELL ABOUT THE PROGRAM
210 CLS
220 PRINT "I CAN CREATE A CHARACTER FOR"
230 PRINT
240 PRINT "  DUNGEONS AND DRAGONS (DD)"
250 PRINT "  RUNEQUEST           (RQ)"
260 PRINT "  TUNNELS AND TROLLS   (TT)"
270 PRINT
280 INPUT "WHICH DO YOU WANT (DD,RQ, OR TT)"; GAME$

300 REM**CHECK OUT THE VALUE OF GAME$
310 IF GAME$ = "DD" THEN CH$ = DD$ : GOTO 410
320 IF GAME$ = "RQ" THEN CH$ = RQ$ : GOTO 410
330 IF GAME$ = "TT" THEN CH$ = TT$ : GOTO 410
340 PRINT "I DON'T UNDERSTAND" GAME$ : GOTO 270

400 REM**ROLL THE CHARACTER
```

Your turn. You do this part.

```
500 REM**ASK IF SOMEONE WANTS ANOTHER CHARACTER
510 PRINT
520 PRINT "FOR ANOTHER CHARACTER, PRESS THE SPACE BAR";
530 K$ = INKEY$ : IF K$ = "" THEN 530
540 IF K$ = " " THEN 210 ELSE 530

999 END
```

Here are some hints.

- For each game, the characteristics are "packed" into a string, DD$ or RQ$ or TT$, in lines 140, 150 and 160.
- In lines 310 through 330, the string CH$ is set equal to one of the strings DD$, RQ$ or TT$.
- Each characteristic occupies exactly three character positions in CH$.
- So, the length of CH$ will be either 18 (for D&D or T&T) or 21 (for Runequest).

We hope these hints helped!

Wandering Star

Wandering Star wanders about the universe looking for cosmic oases that contain her favorite food, cosmic dust. When she find a cosmic oasis, she wanders about it, eating any cosmic dust she finds.

A cosmic oasis looks just like the screen of the TRS-80; cosmic dust sometimes looks like points (•) scattered about the screen. Wandering Star first appears near the center of the screen (oops! cosmic oasis), then rests for a brief time, perhaps thinking hungrily about cosmic dust.

After her brief rest, Wandering Star wanders ... up, down, left, right ... in the oasis. If she should meander into a place that contains a cosmic dust mote, she eats it, then moves on.

Eventually, Wandering Star may reach the edge of the oasis and disappear. This usually happens long before she has eaten all the cosmic dust.

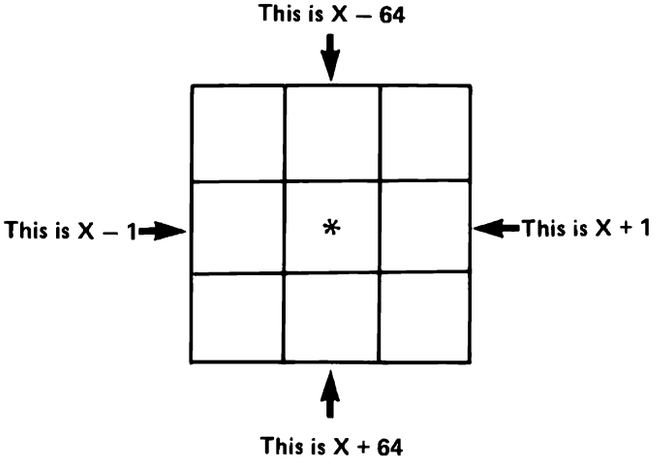
```
100 REM**WANDERING STAR
110 CLS

200 REM**COSMIC DUST
210 FOR K = 1 TO 200
220   PRINT @RND(1022), ".";
230 NEXT K

300 REM**WANDERING STAR APPEARS
310 X = 480
320 PRINT @X, "*";

400 REM**WANDERING STAR RESTS
410 T = 2000
420 FOR Z = 1 TO T : NEXT Z

500 REM**SHE CAN WANDER RIGHT, LEFT,
    DOWN, UP
510 W = RND(4)
520 IF W=1 THEN X2 = X + 1
530 IF W=2 THEN X2 = X - 1
540 IF W=3 THEN X2 = X + 64
550 IF W=4 THEN X2 = X - 64

600 REM**WANDERING STAR WANDERS
610 PRINT @X, " ";
620 X = X2
630 PRINT @X, "*";
640 T = 200
650 FOR Z = 1 TO T : NEXT Z
660 GOTO 510
```

Enter the program and run it. Cosmic dust appears (lines 210 through 230). Then Wandering Star appears (lines 310 and 320) and rests for a few seconds (lines 410 and 420). Wandering Star is at print position X on the screen.

```
320 PRINT @X, "*";
```

Screen position — Wandering Star

Lines 510 through 550 compute a new screen position (X2) for Wandering Star. The new position is chosen at random from four possibilities. The following diagram shows Wandering Star in position X and the four possible places for X2.

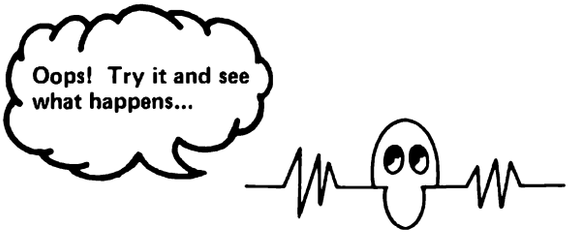


REMEMBER: There are 1024 print positions on the screen, numbered from 0 to 1023. These print positions are arranged in 16 lines with 64 positions in each line.

Aha! That's why we add 64 to X to move *down* one line or subtract 64 to move *up* one line. Again, we suggest you get some "TRS-80 Video Display Worksheets" from your Radio Shack store to help you plan stuff on the screen.

And so we arrive at line 610. This erases Wandering Star from where she now is in screen position X. Then, in line 620, the value of X is replaced by her *new* position X2. Line 630 then prints her in the new position. We will let *you* figure out why we didn't do it as follows.

```
610 PRINT @X, " ";

620 PRINT @X2, "*";
```



Trouble! If X is less than 0 or more than 1023, the TRS-80 will stop with an FC ERROR IN 630. Here is one way to fix that; we will try other ways later. Add the following statements to the program.

```
560 IF X2<0 OR X2>1022 THEN 710

700 REM**WANDERING STAR DISAPPEARS
710 PRINT @X, " ";
720 PRINT @0, "FAREWELL, WANDERING STAR"
730 GOTO 730
```

Now, if Wandering Star wanders out into the cosmic desert, the TRS-80 simply bids her farewell. You may wonder why we used 1022 instead of 1023 in line 560. We will usually avoid using print position 1023 because printing something there causes the entire screen to scroll up one line.

Return of Wandering Star

Well, Wandering Star wanders and ... eventually ... wanders off the screen, never again to appear on-screen. Farewell, Wandering Star.

Alas, the part of the universe surrounding the screen is a cosmic desert. The screen, of course, is a cosmic oasis.

So, after wandering in the desert for awhile, Wandering Star decides to return to the oasis where she can again savor cosmic dust and think about other oases elsewhere in the universe (and therein lies another story).

Think about how Wandering Star might return. She left the oasis along one of the edges of the screen ... hmmm, perhaps she could reappear somewhere at the edge of the screen.

Does she learn from experience? Will she soon disappear again into the desert? Or will she remain in the food-rich oasis, pondering the greater universe, then invent or discover a way to move beyond the cosmic desert into other (and different) oases?

This reminds us of the ancient tale about the inquisitive Russian ruler who watched the night sky, attempting to understand the inexplicable movements of an unpredictable heavenly body.

Yes, Wondering Tsar saw Wandering Star.

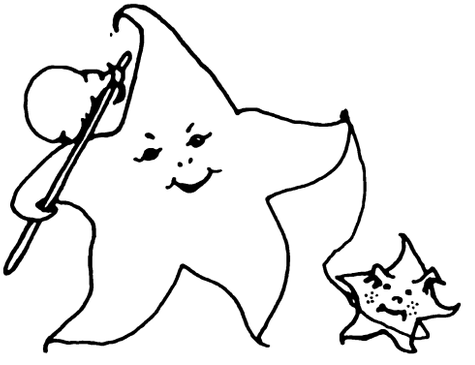
It's easy to keep Wandering Star from leaving the cosmic oasis. Simply add **one** line to the original program as shown below. We have added line 615 (and deleted lines 560 and 700 to 730).

```
600 REM**WANDERING STAR WANDERS
610 PRINT @X, " ";

615 IF X2<0 OR X2>1022 THEN 510

620 X = X2
630 PRINT @X, "*";
640 T = 200
650 FOR Z = 1 TO T : NEXT Z
660 GOTO 510
```

Hmmm ... maybe the following is even better. Instead of line 615, let's put the same statement at line 560.

```
500 REM**SHE CAN WANDER RIGHT, LEFT,
    DOWN OR UP
510 R = RND(4)
520 IF R=1 THEN X2 = X + 1
530 IF R=2 THEN X2 = X - 1
540 IF R=3 THEN X2 = X + 64
550 IF R=4 THEN X2 = X - 64

560 IF X2<0 OR X2>1022 THEN 510
```

Try both of the preceding changes (one at a time, please!). With the first change (line 615), Wandering Star may occasionally disappear briefly (blink, she's gone ... blink, she's back) from the edge of the screen. The other change (line 560) will keep her always in the cosmic oasis. Sometimes she may bump along the edge of the oasis (curiously?), then move away.



Now, change the original program as follows:

```
600 REM**WANDERING STAR WANDERS
610 IF X>=0 AND X<1023 THEN PRINT @X, " ";
620 X = X2
630 IF X>=0 AND X<1023 THEN PRINT @X, "*";
640 T = 200
650 FOR Z = 1 TO T : NEXT Z
660 GOTO 510
```

Try the program with this change. Probably, Wandering Star will reach the edge of the oasis and disappear into the desert. Wait awhile ... she might return. If she doesn't return, press BREAK, then run the program again. Try this several times. Sometimes she returns; sometimes she doesn't return.

To increase the chance that she returns, add this statement:

```
560 IF X2<-100 OR X2>1200 THEN 510
```

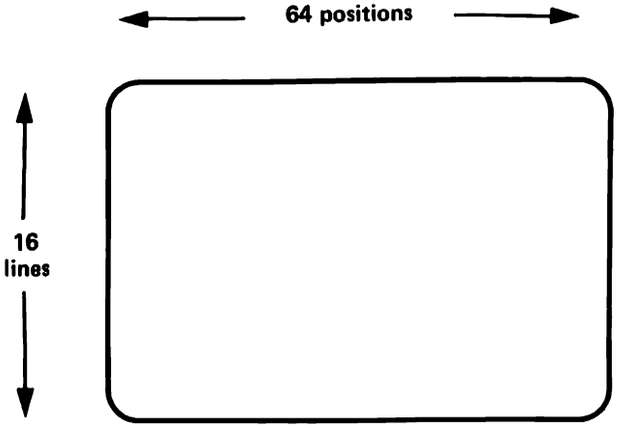
Or, instead of line 560 as shown above, add these three statements:

```
560 L = -100
570 U = 1200
580 IF X2<L OR X2>U THEN 510
```

EXPERIMENT with various values of L and U. What if L = 200 and U = 1000? Try L = −1000 and U = 2000. Try some other values. While you are experimenting, you may wish to reduce the time delay by changing line 640.

Rectangular Oasis

Wandering Star wanders right, left, down or up with equal probability. When she goes off-screen, she usually leaves along the top or bottom edge. Why? Well, remember that the screen has 1024 print positions, arranged in 16 lines with 64 positions in each line.



Hmmm ... perhaps we should teach Wandering Star to wander left or right more, and up or down less. How much more? Of course! Four times as much, on the average, since 64 is 4 times 16.

We can do this by rewriting only block 500 (lines 500 on) in the program, as follows:

```
500 REM**SHE CAN WANDER RIGHT, LEFT,
    DOWN OR UP
510 W = RND(10)
520 IF W<5 THEN X2 = X + 1
530 IF W>6 THEN X2 = X - 1
540 IF W=5 THEN X2 = X + 64
550 IF W=6 THEN X2 = X - 64
```

It works like this:

- In line 510, W is a random number, 1 to 10.
- If W is 1, 2, 3 or 4, then Wandering Star wanders to the right because the condition W<5 in line 520 is true.
- If W is 7, 8, 9 or 10, then she wanders to the left because the condition W>6 in line 530 is true.
- If W = 5, she wanders down because the condition W = 5 in line 540 is true.
- If W = 6, she wanders up because the condition W = 6 in line 550 is true.

You may prefer writing lines 520 through 550 as follows:

```
520 IF W<=4 THEN X2 = X + 1
530 IF W>=7 THEN X2 = X - 1
540 IF W=5  THEN X2 = X + 64
550 IF W=6  THEN X2 = X - 64
```

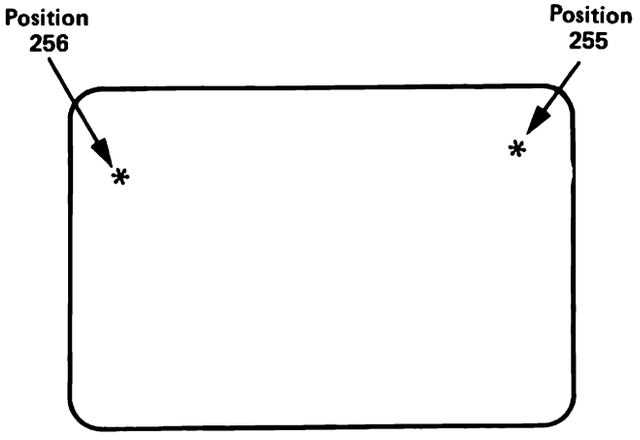
With either of the above, Wandering Star wanders as shown in the following table.

W	She Wanders
1	Right
2	Right
3	Right
4	Right
5	Down
6	Up
7	Left
8	Left
9	Left
10	Left

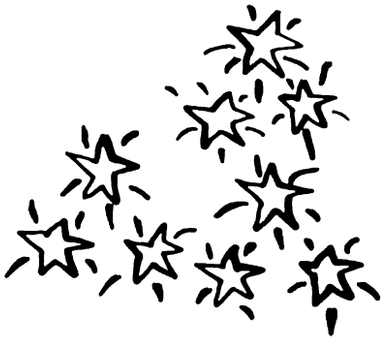
Aha! 4 ways to go right, 4 ways to go left, 1 way to go down, 1 way to go up.

If you now try our Wandering Star program, perhaps you notice that occasionally she wanders in a very strange way. Sometimes she jumps all the way from the left edge of the screen to the right edge, or from the right edge to the left edge. This happens, for example, if she is poised at the right edge of the screen at position 255 and tries to wander one place to the right to position 256. Position 256 is on the *left* edge of the screen, one line down from the line which includes position 255.

Similarly, if she is on the left edge of the screen at position 256 and tries to wander one place to the left, she will appear at position 255 on the right edge of the screen.



This type of "hyperspace" behavior also occurs in many arcade games when a space ship or missile leaves one side of the screen and reappears on the other side. If it annoys you, try your hand at modifying the program so that Wandering Star does not do hyperspace jumps.



She Gets Your Help

As you watch Wandering Star meander about the cosmic oasis, have you ever wished you could help her in her quest for cosmic dust? Well, with the following program, you can. However, the universe also has cosmic noise. So, Wandering Star does not always hear you when you try to help her.

```
100 REM**WANDERING STAR
110 CLS

200 REM**COSMIC DUST
210 FOR K = 1 TO 200
220   PRINT @RND(1022), ".";
230 NEXT K


300 REM**WANDERING STAR APPEARS
310 X = 480 : X2 = X
320 PRINT @X, "*";

400 REM**WANDERING STAR RESTS
410 T = 2000
420 FOR Z = 1 TO T : NEXT Z

500 REM**DOES SHE HEAR US?
510 P = 50
520 IF RND(100)<=P THEN GOSUB 710
    ELSE GOSUB 810

600 REM**WANDERING STAR WANDERS
610 PRINT @X, " ";
620 X = X2
630 PRINT @X, "*";
640 T = 200
650 FOR Z = 1 TO T : NEXT Z
660 GOTO 520

700 REM**SUBROUTINE. SHE HEARS US!
710 K$=INKEY$ : IF K$="" THEN RETURN
720 IF K$ = "R" THEN X2 = X + 1
730 IF K$ = "L" THEN X2 = X - 1
740 IF K$ = "D" THEN X2 = X + 64
750 IF K$ = "U" THEN X2 = X - 64
760 RETURN

800 REM**SUBROUTINE. SHE DIDN'T
    HEAR US
810 W = RND(4)
820 IF W=1 THEN X2 = X + 1
830 IF W=2 THEN X2 = X - 1
840 IF W=3 THEN X2 = X + 64
850 IF W=4 THEN X2 = X - 64
860 RETURN
```

When you run this program, press keys to tell Wandering Star to go Down, Up, Right or Left. Guide her towards cosmic dust.

- Press the R key to tell her to go Right
- Press the L key to tell her to go Left
- Press the D key to tell her to go Down
- Press the U key to tell her to go Up

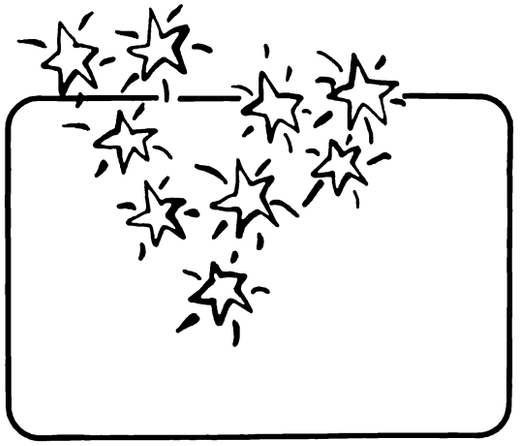
She might hear you, or she might not. This happens in lines 510 and 520. In line 510, we have set P equal to 50. This will cause Wandering Star to hear you about 50% of the time. If you change P to 30, she will hear you about 30% of the time. Try 70. She will hear you about 70% of the time. If you change P to 100, she will hear you all the time. EXPERIMENT: Try different values for P in line 510.

The value of P is the probability (in percent) that Wandering Star will go in the direction of your key press (R, L, D or U) instead of wandering randomly. Why? Look at line 520.

```
520 IF RND(100)<=P THEN GOSUB 710 ELSE GOSUB 810
```

Aha! RND(100) is a random integer in the range 1 to 100. Suppose P = 50. About 50% of the time, RND(100) will be less than or equal to 50. In this case, the condition RND(100)<=P is *true* and the TRS-80 will GOSUB 710. About 50% of the time, RND(100) will be greater than 50 and the condition RND(100)<=P will be *false*. In this case, the TRS-80 will obey the ELSE clause and GOSUB 810.

This program does not prevent Wandering Star from trying to wander off the screen. How can you help keep her on screen and moving towards delicious cosmic dust? Also, how can you make Wandering Star's diet more interesting? After all, would *you* like to eat *only* cosmic dust for millions of years?



Gourmet Oasis

The universe is incredibly large and varied. So, Wandering Star might occasionally find a gourmet oasis. We offer the following gourmet delights for Wandering Star.

Instead of no-tailed cosmic dust, how about some long-tailed cosmic dust.

```
200 REM**LONG-TAILED COSMIC DUST
210 FOR K = 1 TO 200
220   PRINT @RND(1022), ",";
230 NEXT K
```

An oasis might have both kinds (• and ❟).

```
200 REM**TWO KINDS OF COSMIC DUST
210 FOR K = 1 TO 200
220   R = RND(2)
230   IF R=1 THEN FOOD$ = "."
240   IF R=2 THEN FOOD$ = ","
250   PRINT @RND(1022), FOOD$;
260 NEXT K
```

And now — the gourmet oasis featuring escargot (@) sprinkled in with no-tailed (•) and long-tailed (❟) dust.

```
200 REM**THE GOURMET OASIS
210 MENU$ = "........,,@"
220 FOR K = 1 TO 200
230   R = RND(10)
240   FOOD$ = MID$(MENU$, R, 1)
250   PRINT @RND(1022), FOOD$;
260 NEXT K
```

Since MENU$ consists of seven no-tailed dust motes (•), two long-tailed dust motes (❟) and one escargot (@), you will see them in the oasis in approximately a 7 to 2 to 1 ratio. You can change the mix or even change the available gourmet delights by modifying lines 210 and 230.

Perhaps she discovers an alphabet soup oasis. Change lines 210 and 230, as follows:

```
210 MENU$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

230 R = RND(26)
```

Or, write line 230 as follows and you can change only the MENU$ to get a new assortment of food.

```
230 R = RND(LEN(MENU$))
```

Programming Problems

Your turn. Try your hand at these problems, which were first published in the March-April 1980 issue of *Recreational Computing* Magazine. The problems are numbered as published in RC.

PROBLEM 10 1980

This one is easy. Complete line 20 in the following BASIC program so that the computer will print the number of the year 1980.

```
10 CLS

20 YEAR = _____

30 PRINT YEAR
40 END
```

Wait! There are a few rules.

(1) Your BASIC expression must include every nonzero digit: 1 2 3 4 5 6 7 8 9

(2) Every nonzero digit must appear once, and only once, in line 20.

(3) You may not use 0.

It can be done. Here are some examples

```
20 YEAR=4*5*9*(8+3)*(7+1)/(6+2)

20 YEAR=45*(89-67)*(3+1)/2

20 YEAR=198*(7+3)*(4+5-2-6)
```

The above examples are quite mundane. Dwellers of Xanth and other magical places will use the full power of BASIC to generate the number 1980.

PROBLEM 11 WORD'S WORTH

Assign numbers to the letters A through Z, as follows:

A = 1	G = 7	M = 13	S = 19	Y = 25
B = 1	H = 8	N = 14	T = 20	Z = 26
C = 3	I = 9	O = 15	U = 21	
D = 4	J = 10	P = 16	V = 22	
E = 5	K = 11	Q = 17	W = 23	
F = 6	L = 12	R = 18	X = 24	

A word's worth is its numerical value, obtained by adding the values of the letters in the word. For example, HOBBIT is worth 56 points, DRAGON is worth 59 points and WIZARD is worth 81 points.

Write a program to compute a word's worth. A RUN of your program might look like this:

```
YOUR WORD? WIZARD
YOUR WORD IS WORTH 81 POINTS

YOUR WORD? ISN'T
YOUR WORD IS WORTH 62 POINTS

YOUR WORD? FLIP-FLOP
YOUR WORD IS WORTH 92 POINTS

YOUR WORD? 3#AB%*Z
I DON'T UNDERSTAND
```

And so on. Your program should compute the worth of any word or even any string of letters, even if it isn't a word. Contractions and hyphenated words are OK. Your program should accept strings that include:

Letters, A through Z
Apostrophes (')
Hyphens (-)
Spaces

THIRTEEN is worth 99 points. Are there numbers whose English name has a word's worth equal to the number? If so, what is the smallest such number?

AHA is worth 10 points, AHA is a palindrome, but . . . sigh . . . 10 is not a palindrome. Are there words which are palindromes for which the word's worth is also a palindrome?

PROBLEM 14 EXOTIC NAMES IN FANTASY ADVENTURELAND

Suppose you are creating a fantasy adventure for *Dungeons and Dragons*, *Runequest*, or *Tunnels and Trolls*. You may wish to use unusual names for your heroes, wizards, monsters and other creatures. You could borrow names from books such as *Lord of the Rings*. But, perhaps you prefer to invent your own.

Why not use your computer to help you invent names? Sounds OK, but how do you get the computer to print names that are pronounceable (or almost so) and seem to be unusual, exotic or even fantastic?

That's the problem. Write a program to generate random names that might be used in a fantasy or science fiction game or story.

Yes, this does sound more like a *project* than a *problem*. Good! We hope you will project your best efforts into this problem . . . er, project . . . and devise some cunningly contrived programs to generate fantastic names. Let's see now — how many orcs would have to type for how many years to write *Lord of the Rings?*